Progetto di Sistemi Distribuiti e Pervasivi

Laboratorio di Sistemi Distribuiti e Pervasivi

Maggio 2015

1 Descrizione del progetto

Lo scopo del progetto è quello di realizzare un sistema di monitoraggio remoto di un ambiente sensorizzato simulato. I sensori simulati facenti parte dell'ambiente devono coordinarsi tra di loro per comunicare le loro misurazioni ad un gateway, il quale ha il compito di memorizzarle internamente. Il gateway ha anche il ruolo di offrire ai tecnici del sistema un'interfaccia per effettuare svariati tipi di interrogazioni, in modo da ottenere informazioni sullo stato dell'ambiente (ad esempio la temperatura media degli ultimi n minuti). L'architettura generale del sistema è illustrata in Figura 1. I tre componenti principali da realizzare sono: Rete di sensori, Gestore e Tecnico. La Rete di sensori consiste in un insieme di processi che simulano i diversi tipi di sensore installati nell'ambiente. Questi processi dovranno coordinarsi tra di loro per trasmettere le varie misurazioni a Gestore. Gestore è il server che ha il compito di ricevere e memorizzare i dati provenienti dai sensori, svolgendo il ruolo di gateway. Inoltre, deve anche predisporre un sistema di monitoraggio da remoto che permetta di effettuare diversi tipi di interrogazioni sullo stato del sistema. Le interrogazioni vere e proprie vengono effettuate dai tecnici tramite l'apposito client Tecnico.

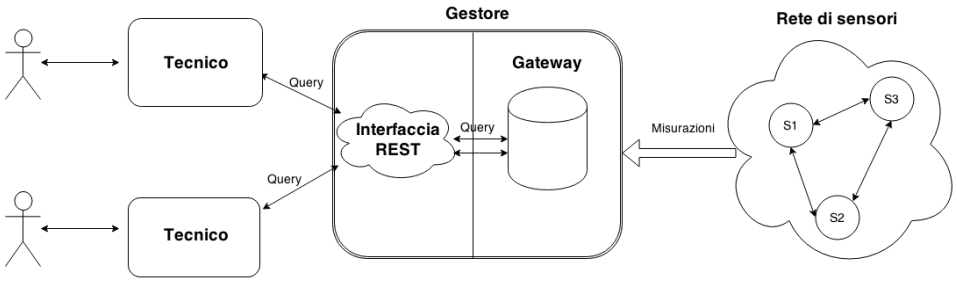

Figura 1: Architettura del sistema.

Le prossime sezioni di questo documento descriveranno in dettaglio le componenti del sistema da implementare.

2 Rete di sensori

L'ambiente simulato da monitorare è una stanza dotata di quattro sensori:

- Un sensore che periodicamente misura la temperatura della stanza in gradi Celsius
- Un sensore che periodicamente misura la luminosità della stanza in lux
- Un sensore di presenza PIR che memorizza un evento ogni qualvolta viene rilevata la presenza di una persona all'interno della zona est della stanza
- Un sensore di presenza PIR che memorizza un evento ogni qualvolta viene rilevata la presenza di una persona all'interno della zona ovest della stanza

Ognuno di questi sensori deve essere simulato da un relativo **processo**, che chiameremo nodo. Ci saranno quindi in tutto quattro **processi**, uno per tipologia di sensore. I nodi hanno la possibilità di comunicare con gli altri nodi e con Gestore tramite socket. Ogni nodo ha un suo livello di batteria iniziale, che viene decrementato in seguito a letture di misurazioni e alla trasmissione di messaggi. All'interno della rete, viene eletto dinamicamente (in base ai livelli di batteria) un particolare nodo: il sink. Quest'ultimo ha il compito di richiedere periodicamente le misurazioni più recenti agli altri nodi della rete per poi trasmetterle (insieme alle sue) a Gestore.

2.1 Misurazioni

Ogni misurazione di sensore è caratterizzata da:

- Id del sensore
- Valore letto
- Timestamp in millisecondi

La generazione di queste misurazioni viene svolta da opportuni simulatori, che vengono forniti già pronti per semplificare lo svolgimento del progetto. Ogni simulatore assegna come timestamp alle misurazioni il numero di millisecondi passati da quando è partito.
Al link *http://ewserver.di.unimi.it/sdp/simulation_src.zip* è quindi possibile trovare il codice necessario. Questo codice andrà aggiunto come package al progetto e NON deve essere modificato.

Per ogni tipologia di sensore, viene fornita una classe che implementa un

thread di simulazione. Ogni nodo deve quindi occuparsi di far partire il thread giusto in base al ruolo che deve svolgere. Ogni thread consiste in un loop infinito che simula periodicamente (con frequenza predefinita) le misurazioni, inserendole in una opportuna struttura dati di capienza limitata. Di questa struttura dati viene fornita solo l'interfaccia (Buffer), la quale espone due metodi:

- *void aggiungi*($Misurazione\ m$)
- *List* <$Misurazione$> *leggi*()

È dunque necessario creare una classe che implementi l'interfaccia: un buffer con capienza massima di 10 misurazioni. Ogni **processo** che simula un sensore avrà un suo buffer. I thread di simulazione usano il metodo *aggiungi* per riempire la struttura dati. Se la capienza massima viene raggiunta, le misurazioni più vecchie vengono sovrascritte con le più recenti. Ogni volta che viene aggiunta una misurazione al buffer, questa è considerabile una lettura per il sensore (il livello di batteria deve quindi essere aggiornato). Il metodo *leggi* deve invece essere usato per ottenere tutte le misurazioni contenute nella struttura dati. Al termine di una lettura, *leggi* deve occuparsi di svuotare il buffer in modo da fare spazio a nuove misurazioni.

2.2 Sink

Per ogni nodo della rete, il costo di comunicazione con il gateway è più alto rispetto a quello con altri nodi. Per ridurre il consumo di batteria, all'interno della rete solo un nodo (detto sink) ha il compito di comunicare direttamente con Gestore. Il sink si occupa quindi periodicamente (in base alla frequenza di trasmissione specificata come parametro) di chiedere agli altri nodi la trasmissione delle loro ultime misurazioni. Una volta ricevute, sarà suo compito quello di trasmetterle (aggiungendo le sue) a Gestore. Inoltre, se il sink si accorge che un nodo della rete ha esaurito la batteria, deve avvisare Gestore con un opportuno messaggio di errore. Si assume che un nodo con la batteria scarica chiuda le comunicazioni in entrata. In fase di inizializzazione della rete, il ruolo del sink viene assegnato di default (tramite argomenti) ad un particolare nodo. Se il sink si accorge di avere un livello di batteria inferiore al 25%, proverà a contattare gli altri nodi per passare il compito a qualcuno con un più alto livello di batteria. Può essere eletto solamente un nodo con livello di batteria superiore al 25%. Se nessun nodo può essere eletto, il nodo con più alto livello di batteria deve comunicare a Gestore un messaggio di rete non disponibile. È quindi necessario formalizzare (realizzando lo schema di comunicazione) ed implementare un protocollo che realizzi un opportuno algoritmo di elezione. Questo protocollo deve fare in modo che solo il nodo con più alto livello di batteria venga eletto. In fase di discussione del progetto, dovrà essere mostrato e spiegato lo schema di comunicazione del protocollo realizzato.

2.3 Livello batteria

Dato t il costo di trasmissione e l il costo di lettura di misurazione, il livello di batteria b deve essere aggiornato nel seguente modo:

$$b_{new} = \begin{cases} b_{old} - t, & \text{se viene trasmesso un messaggio ad un altro sensore} \\ b_{old} - 4t, & \text{se viene trasmesso un messaggio al gateway} \\ b_{old} - l, & \text{se viene letto un dato} \end{cases}$$

Un nodo dovrà chiudere le comunicazioni verso l'interno e terminare nel caso in cui il suo livello di batteria non gli permetta di effettuare una di queste operazioni. Per semplicità, se un nodo n_1 ha la batteria scarica ma un altro nodo n_2 è in attesa di una suo messaggio, n_1 finisce di gestire le comunicazioni in sospeso prima di terminare. Supponiamo ad esempio che un nodo abbia ricevuto con successo una richiesta di misurazioni dal sink, ma che nel frattempo la sua batteria si sia scaricata. In questo caso, il nodo risponderà al sink in modo tale che quest'ultimo non rimanga in perenne attesa di una sua risposta.

2.4 Inizializzazione rete di sensori

Per essere inizializzata, la Rete di Sensori necessita dei seguenti parametri:

- L'id del nodo che inizierà come *sink*
- La frequenza di trasmissione del *sink* (in millisecondi)
- Il livello di batteria di ogni sensore

2.5 Inizializzazione *nodo*

Ogni singolo **processo** che implementa un *nodo* necessita di diversi argomenti per essere avviato:

- Tipologia di sensore (temperatura, luminosità o pir)
- Livello iniziale di batteria
- Un valore booleano per stabilire se il nodo inizierà con il ruolo di sink
- La frequenza di trasmissione del *sink* (in millisecondi)

I seguenti parametri, invece, saranno fissi

- Livello di batteria massimo: 1000
- Costo di trasmissione: 5
- Costo di lettura di una misurazione: 2

3 Gestore

L'applicazione Gestore è incaricata di svolgere il ruolo di gateway per la rete di sensori, memorizzando le misurazioni acquisite dal sink internamente in un'opportuna struttura dati. Gestore espone inoltre ai tecnici un'interfaccia REST, che permette loro di realizzare le interrogazioni necessarie ad ottenere informazioni sullo stato del sistema. Queste interrogazioni vengono quindi realizzate sulla struttura dati che contiene lo stream di misurazioni provenienti dalla rete di sensori. Infine, si occupa di inviare ad ogni tecnico registrato al sistema i problemi derivanti dalla rete (come ad esempio la batteria scarica di un nodo).

3.1 Interfaccia con i tecnici

L'applicazione Gestore mette a disposizione dei tecnici un servizio REST per effettuare le interrogazioni sui dati collezionati dai sensori. Devono quindi essere disponibili diversi tipi di servizi:

- Log in di un tecnico
- Interrogazione della rete di sensori
- Log out di un tecnico

Per effettuare il log in, un tecnico deve semplicemente inviare il proprio nome utente che deve essere univoco all'interno del sistema. Nel caso che il nome trasmesso da un tecnico sia già stato usato, deve essere trasmesso un opportuno messaggio di errore.

3.2 Interfaccia con i sensori

Un altro compito dell'applicazione Gestore è quello di gestire il gateway che riceve (tramite socket) e memorizza in un'opportuna struttura dati (che per semplicità può essere in memoria centrale) le misurazioni derivanti dalla rete di sensori. Questo modulo si occupa anche di gestire i messaggi contenenti problemi tecnici derivanti dal sink (ad esempio un messaggio di batteria scarica) e di comunicarli a tutti i tecnici registrati al sistema.

3.3 Interrogazioni

Il sistema deve essere predisposto a gestire diversi tipi di interrogazione:

- Per temperatura e luminosità:
  - Ottenere la misurazione più recente e il relativo timestamp
  - Ottenere la media delle misurazioni dal tempo t_1 al tempo t_2
  - Ottenere minimo e massimo dal tempo $t1$ al tempo $t2$

- Ottenere la luminosità dell'istante di tempo più vicino a quello in cui la temperatura ha raggiunto un massimo tra l'istante t_1 e l'istante t_2
- Stabilire quante volte è stata rilevata la presenza di una persona dal t_1 al tempo t_2:
  - Nella zona ovest
  - Nella zona est
  - In media tra le due zone

Nel caso di interrogazioni che agiscono su intervalli, deve essere restituito un opportuno messaggio di errore se non sono presenti dati nell'intervallo specificato. Allo stesso modo, nel caso di interrogazioni che richiedono misurazione più recente, deve essere restituito un messaggio di errore se quel tipo di misurazioni non è ancora stato collezionato.

4 Tecnico

L'applicazione Tecnico è un client (per semplicità a linea di comando) utilizzato dai tecnici che intendono monitorare il sistema. In fase iniziale vengono richiesti all'utente il nome utente che il tecnico intende utilizzare e l'indirizzo (ip e porta) del sistema da monitorare. Con queste informazioni, il programma si deve occupare di registrare il tecnico al sistema. Se viene ricevuto un messaggio dal sistema di nome utente già utilizzato, deve essere chiesto all'utente di inserirlo nuovamente. A registrazione completata correttamente, il programma propone in loop all'utente una scelta:

- Effettuare una delle interrogazioni specificate nella sezione precedente
- Terminare il programma (in questo caso il programma dovrà segnalare il log out del tecnico al sistema)

L'applicazione Tecnico, inoltre, deve restare in ascolto di comunicazioni di problemi tecnici da parte del sistema (ad esempio, batterie dei sensori scariche). Quando un messaggio di questo tipo viene ricevuto, deve essere comunicato tempestivamente all'utente.

5 Semplificazioni e limitazioni

Si ricorda che lo scopo del progetto è dimostrare la capacità di progettare e realizzare un'applicazione distribuita e pervasiva. Pertanto gli aspetti non riguardanti il protocollo di comunicazione, la concorrenza e la gestione dei dati di sensori sono considerati secondari. Inoltre è possibile assumere che :

- nessun nodo si comporti in maniera maliziosa,

- nessun nodo termini in maniera incontrollata

Si gestiscano invece i possibili errori di inserimento dati da parte dell'utente.

Sebbene le librerie di Java forniscano molteplici classi per la gestione di situazioni di concorrenza, per fini didattici gli studenti sono invitati a fare esclusivamente uso di metodi e di classi spiegati durante il corso di laboratorio. Pertanto, eventuali strutture dati di sincronizzazione necessarie (come ad esempio semafori o buffer condivisi) dovranno essere implementate da zero e saranno discusse durante la presentazione del progetto.

Nonostante alcune problematiche di sincronizzazione possano essere risolte tramite l'implementazione di server iterativi, per fini didattici si richiede di utilizzare server multithread.

6 Presentazione del progetto

Il progetto è da svolgere individualmente. Durante la valutazione del progetto verrà richiesto di mostrare alcune parti del programma, verrà verificata la padronanza del codice presentato, verrà verificato il corretto funzionamento del programma e verranno inoltre poste alcune domande di carattere teorico inerenti gli argomenti trattati nel corso (parte di laboratorio). E' necessario presentare il progetto sul proprio computer.

Il codice sorgente dovrà essere consegnato al docente prima della discussione del progetto. Per la consegna, è sufficiente archiviare il codice in un file zip, rinominato con il proprio numero di matricola (es. 654321.zip) ed effettuare l'upload dello stesso tramite il sito http://upload.dico.unimi.it . Sarà possibile effettuare la consegna a partire da una settimana prima della data di ogni appello. La consegna deve essere tassativamente effettuata entro le 23:59 del secondo giorno precedente quello della discussione (es. esame il 13 mattina, consegna entro le 23.59 dell'11). Eventuali modifiche apportate al progetto dopo il termine della consegna devono essere segnalate al docente prima della discussione.

Si invitano inoltre gli studenti ad utilizzare, durante la presentazione, le istruzioni `Thread.sleep()` al fine di mostrare la correttezza della sincronizzazione del proprio programma. Si consiglia vivamente di analizzare con attenzione tutte le problematiche di sincronizzazione e di rappresentare lo schema di comunicazione fra le componenti. Questo schema deve rappresentare il formato dei messaggi e la sequenza delle comunicazioni che avvengono tra le componenti in occasione delle varie operazioni che possono essere svolte. Tale schema sarà di grande aiuto, in fase di presentazione del progetto, per verificare la correttezza della parte di sincronizzazione distribuita.

Nel caso in cui il docente si accorga che una parte significativa del progetto consegnato non è stata sviluppata dallo studente titolare dello stesso,

lo studente in causa: a) dovrà superare nuovamente tutte le prove che compongono l'esame del corso. In altre parole, se l'esame è composto di più parti (teoria e laboratorio), lo studente dovrà sostenere nuovamente tutte le prove in forma di esame orale (se la prova di teoria fosse già stata superata, questa verrà annullata e lo studente in merito dovrà risostenerla). b) non potrà sostenere nessuna delle prove per i 2 appelli successivi. Esempio: supponiamo che gli appelli siano a Febbraio, Aprile, Giugno, Luglio, e che lo studente venga riconosciuto a copiare all'appello di Febbraio. Lo studente non potrà presentarsi agli appelli di Aprile e Giugno, ma potrà sostenere nuovamente le prove dall'appello di Luglio in poi. Il docente si riserva la possibilità di assegnare allo studente in causa lo svolgimento di una parte integrativa o di un nuovo progetto.

7 Parti facoltative

7.1 Prima parte

I sensori hanno solitamente a disposizione poco spazio per memorizzare le misurazioni raccolte. Implementare quindi un sistema di compressione dei dati a bordo del sensore (tra quelli visti a lezione), in modo tale da risparmiare spazio di memorizzazione. Per questa parte facoltativa la capacità del buffer all'interno dei nodi deve essere allargata a 100, in modo da testare l'algoritmo di compressione con una quantità ragionevole di misurazioni.

7.2 Seconda parte

Una delle assunzioni fatte è che un nodo che ha scaricato la batteria può comunque rispondere alle ultime richieste in sospeso (ad esempio una richiesta di misurazioni da parte del sink). In questa parte facoltativa quest'assunzione deve essere rimossa: un nodo che ha scaricato la batteria deve chiudere tutte le comunicazioni e terminare anche se qualche altro nodo della rete si aspetta una sua risposta. È quindi necessario trovare una soluzione (giustificandola adeguatamente in fase di discussione del progetto) per permettere alla Rete di Sensori di "capire" quando una risposta da un nodo non arriva perchè la sua batteria è scarica, evitando di rimanere in perenne attesa.

Nota bene: anche l'algoritmo di elezione del sink dovrà tenere conto di nodi che possono smettere di rispondere!

8 Aggiornamenti

Qualora fosse necessario, il testo dell'attuale progetto verrà aggiornato al fine di renderne più semplice l'interpretazione. Le nuove versioni del progetto

verranno pubblicate sul sito del corso. Si consiglia agli studenti di controllare regolarmente il sito.

Al fine di incentivare la presentazione del progetto nei primi appelli disponibili, lo svolgimento della parte facoltativa 1 diventa obbligatoria a partire dall'appello di Settembre (incluso), mentre entrambe le parti facoltative (1 e 2) saranno obbligatorie negli appelli successivi.